ENAE 441 - 0101

HW03: System Propagation

Due on October 30th. 2025 at 09:30 AM

Dr. Martin, 09:30 AM

Vai Srivastava

October 30, 2025

Problem 1: Propagation Functions *(30 pts.)*

Please implement the following functions in your code to confirm your understanding of the different propagation strategies. You are not required to use these exact functions in subsequent questions, as there may be more appropriate / efficient implementations for subsequent questions. They are merely provided as a way to confirm your own understanding of the process.

(a) Write a function that propagates a LTI system forward numerically[1] and plot the displacement $x(t)$ and velocity $\dot{x}(t)$.

(b) Write a function that propagates a LTI system forward analytically using a continuous time (CT) STM[2].

(c) Write a function that propagates a LTI system forward analytically using a discrete time (DT) STM[3].

(d) Write a function that propagates a LTV system by jointly integrates the nominal trajectory and the state transition matrix for a fixed amount of time[4].

(e) Write a function that returns the maximum allowable sampling time for a discrete time system to avoid aliasing.

Solution

Part A

```python
# REQUIRED --- 1a
def propogate_CT_LTI_numerically(X_0, X_dot_fcn, t_vec):
    t0 = float(t_vec[0])
    tf = float(t_vec[-1])

    reversed_time = tf < t0
    if reversed_time:
        t_eval = np.asarray(t_vec)[::-1]
        t_span = (float(t_eval[0]), float(t_eval[-1]))
    else:
        t_eval = np.asarray(t_vec)
        t_span = (t0, tf)

    sol = solve_ivp(
        fun=X_dot_fcn,
        t_span=t_span,
        y0=np.asarray(X_0).reshape(-1),
        t_eval=t_eval,
        rtol=1e-9,
        atol=1e-12,
        vectorized=False,
```

[1] using Python's `scipy.integrate.solve_ivp`

[2] Be sure to use `scipy.linalg.expm` to make the matrix exponential

[3] Be sure to use `numpy.linalg.matrix_power` to raise the matrix to a power

[4] Hint: Form a joint state-STM vector $\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x}_{\text{nom}}(t) & \phi(t, t_0) \end{bmatrix}^\top$. Unroll the STM into a single column vector such that the object passed to `solve_ivp` is a single column vector. Use $\boldsymbol{\Phi}(t_0, t_0) = \mathcal{I}_{6\times6}$ for the initial condition

Problem 1 continued on next page...

```python
    )

    if not sol.success:
        raise RuntimeError(f"solve_ivp failed: {sol.message}")

    X_t = sol.y.T
    if reversed_time:
        X_t = X_t[::-1]

    # Return trajectory over time where np.shape(X_t) = (len(t_vec), len(X_0))
    return X_t
```

Part B

```python
# REQUIRED --- 1b
def propogate_CT_LTI_analytically(X_0, A, t_vec):
    X_0 = np.asarray(X_0).reshape(-1)
    A = np.asarray(A)
    t_vec = np.asarray(t_vec)
    t0 = float(t_vec[0])

    X_t = np.empty((len(t_vec), len(X_0)), dtype=float)
    for i, t in enumerate(t_vec):
        Phi = expm(A * (float(t) - t0))
        X_t[i] = Phi @ X_0

    # Return trajectory over time where np.shape(X_t) = (len(t_vec), len(X_0))
    return X_t
```

Part C

```python
# REQUIRED --- 1c
def propogate_DT_LTI_analytically(X_0, A, dt, k_max):
    X_0 = np.asarray(X_0).reshape(-1)
    A = np.asarray(A)
    A_d = expm(A * float(dt))

    n = X_0.size
    X_t = np.empty((int(k_max) + 1, n), dtype=float)

    # x[0]
    X_t[0] = X_0

    # x[k] = A_d^k x[0]
    # (explicitly use matrix_power as the assignment requests)
    for k in range(1, int(k_max) + 1):
        A_dk = np.linalg.matrix_power(A_d, k)
        X_t[k] = A_dk @ X_0
```

```python

    # Return trajectory over time where np.shape(X_t) = (len(t_vec), len(X_0))
    return X_t
```

Part D

```python
# REQUIRED --- 1d
def propagate_LTV_system_numerically(X_0, x_dot_fcn, A_fcn, t_vec):
    n = len(X_0)
    phi0 = np.eye(n).flatten()
    z0 = np.hstack((X_0, phi0))

    def z_dot(t, z):
        x = z[:n]
        phi = z[n:].reshape((n, n))

        x_dot = x_dot_fcn(t, x)
        A_t = A_fcn(x)
        phi_dot = A_t @ phi

        return np.hstack((x_dot, phi_dot.flatten()))

    sol = solve_ivp(z_dot, [t_vec[0], t_vec[-1]], z0, t_eval=t_vec)

    X_t_vec = sol.y[:n, :].T
    phi_t_vec = np.zeros((len(t_vec), n, n))
    for i in range(len(t_vec)):
        phi_t_vec[i] = sol.y[n:, i].reshape((n, n))

    # Return trajectory and STM over time where
    # np.shape(X_t_vec) = (len(t_vec), len(X_0))
    # np.shape(phi_t_vec) = (len(t_vec), len(X_0), len(X_0))

    return X_t_vec, phi_t_vec
```

Part E

```
0.7853981633974481
```

Code

See the Python code for this assignment.

Problem 2: Continuous Time Linear System *(30 pts.)*

Consider a spring-mass-damper system

$$m\,\ddot{x}\,(t) + c\,\dot{x}\,(t) + k\,x\,(t) = 0$$

with system mass $m = 1$, damping coefficient $c = 0.5\,\frac{\mathrm{N\,s}}{\mathrm{m}}$, and spring constant $k = 4\,\frac{\mathrm{N}}{\mathrm{m}}$ and an initial condition of $\boldsymbol{x}\,(0) = \begin{bmatrix}1\\0\end{bmatrix}$.

The system is measured by observing the position

$$y\,(t) = x\,(t)$$

(a) Express the system as a continuous-time state-space model and as a discrete-time model. Define a state vector of $\boldsymbol{x}\,(t) = \begin{bmatrix}x\,(t)\\ \dot{x}\,(t)\end{bmatrix}$ and define all of the corresponding matrices. Assume no external input ($\boldsymbol{u}\,(t) = 0$).

(b) Propagate the system forward for 10 seconds using all three LTI propagation methods and plot $x\,(t)$ and $\dot{x}\,(t)$ on the same figure.

(c) Compare the results. What happens when you apply a Δt value that exceeds the critical threshold in the DT STM?

(d) Use the set of position measurements in `HW3-spring-data.npy` to determine the initial state $\boldsymbol{X}\,(t = 0)$ for a different trajectory.

(e) How many measurements are needed to ensure the state $\boldsymbol{x}\,(t)$ is observable if $\Delta t = 1\,\mathrm{s}$? Explain your findings.

Solution

Part A

With state $x(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix}$ and output $y(t) = x(t)$, the continuous-time model is

$$\dot{x}(t) = \mathbf{A}x(t), \quad \mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -4 & -0.5 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \mathbf{D} = \mathbf{0}$$

For a sampling period Δt, the discrete-time model (no input) is

$$x_{k+1} = \mathbf{A}_d x_k, \quad y_k = \mathbf{C}x_k,$$

with $\mathbf{A}_d = e^{\mathbf{A}\Delta t} = \Phi(\Delta t)$

Because this is a 2nd-order underdamped oscillator, it's convenient to write $\alpha = \frac{c}{2m} = \frac{1}{4}, \omega_n = \sqrt{k/m} = 2$, and $\omega_d = \omega_n\sqrt{1-\zeta^2} = 2\sqrt{1 - \left(\frac{c}{2m\omega_n}\right)^2} = 1.98431348$.

The continuous-time STM is

$$\Phi(t) = e^{-\alpha t}\begin{bmatrix} \cos(\omega_d t) + \frac{\alpha}{\omega_d}\sin(\omega_d t) & \frac{1}{\omega_d}\sin(\omega_d t) \\ -\frac{\omega_n^2}{\omega_d}\sin(\omega_d t) & \cos(\omega_d t) - \frac{\alpha}{\omega_d}\sin(\omega_d t) \end{bmatrix}$$

Hence the discrete state matrix is $\mathbf{A}_d = \boldsymbol{\Phi}(\boldsymbol{\Delta}t)$ with the same $\alpha, \omega_n, \omega_d$ substituted, and $\mathbf{C}_d = \mathbf{C}, \mathbf{D}_d = \mathbf{D} = \mathbf{0}$

Problem 2 continued on next page...

Part B

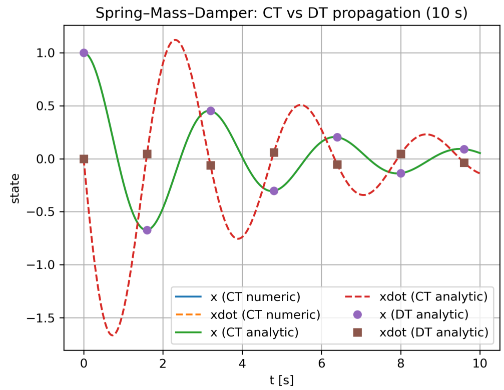

Figure 1: Propagation Trajectories

Part C

```

Comparison:
- CT numerical (solve_ivp) integrates \dot{x} = A x and returns x(t) at requested t. Accuracy depends on tolerances;
  it handles stiff/ill-conditioned cases with adaptive steps.
- CT analytic uses the continuous-time STM Φ(t, t0) = expm(A (t-t0)), giving the exact state at any t for LTI systems.
- DT analytic samples the exact CT solution at multiples of Δt by forming A_d = expm(A Δt) and x[k] = A_d^k x[0].
  At the sample instants, this is mathematically identical to CT analytic.

Critical sampling:
Let λ = σ ± j ω_d be complex poles and ω_n = |λ| the undamped natural frequency. The Nyquist bound is
    Δt_max = π / max ω_n.
For this system, max ω_n = 2 rad/s ⇒ Δt_max = π/2 ≈ 1.5708 s.

What happens if Δt > Δt_max?
- The discrete-time eigenangle is θ = ω_d Δt. Angles are wrapped modulo 2π in discrete time, producing an
  aliased apparent frequency ω_alias = |ω_d - 2π/Δt · round(ω_d Δt / 2π)|.
- The DT trajectory x[k] = A_d^k x[0] is still the exact CT solution *at those sample instants*, but if you attempt to
  infer the underlying oscillation from samples (or connect samples with lines), you will see a lower, incorrect
  frequency (and possibly sign flips when θ ≈ π). Estimation and identification tasks will misinterpret the dynamics.
- Stability is preserved here because |e^{λ Δt}| = e^{σ Δt} < 1 (σ < 0), but phase is misrepresented beyond Nyquist.
```

Part D

```
[[2.99911711]
 [1.69178453]]
```

Part E

```

2 measurements (e.g., at k=0 and k=1 with Δt=1 s) are sufficient.
Reason: the 2×2 observability matrix O₂ = [ C ; C A_d ] equals
    [[1, 0],
     [Φ₁₁(Δt), Φ₁₂(Δt)]],
which has full rank iff Φ₁₂(Δt) ≠ 0. For this system,
    Φ₁₂(Δt) = e^(-α Δt) · (1/ω_d) · sin(ω_d Δt),
and with α=0.25, ω_d≈1.9843 rad/s, Δt=1 s ⇒ sin(ω_d Δt) ≠ 0, so rank(O₂)=2.
Thus the discrete-time state is observable from two consecutive position measurements.
```

Code

See the Python code for this assignment.

Problem 3: Continuous Time Non-Linear System *(40 pts.)*

Consider a satellite in a 3D Keplerian orbit around Earth. The dynamics of the satellite can be modeled using Newton's second law under the gravitational force:

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r},$$

where $\mu = 398\,600\,\frac{\mathrm{km}^3}{\mathrm{s}^2}$ is the Earth's gravitational parameter and $\boldsymbol{r}$ is the position vector of the satellite in the Earth-centered inertial (ECI) frame.
The initial position and velocity vectors are:

$$^{\mathcal{N}}\boldsymbol{X}(0) = \begin{bmatrix} 7000\,\mathrm{km} \\ 0.0\,\mathrm{km} \\ 0.0\,\mathrm{km} \\ 0.0\,\frac{\mathrm{km}}{\mathrm{s}} \\ 7.5\,\frac{\mathrm{km}}{\mathrm{s}} \\ 3.5\,\frac{\mathrm{km}}{\mathrm{s}} \end{bmatrix}^\top$$

and the measurement function of the system is the range, ρ taken from an observer at $\phi = 30\,^\circ$ latitude and $\lambda = 60\,^\circ$ longitude, where the radius of the Earth, R_E is taken as $6378\,\mathrm{km}$ and the rotation of the Earth is $\omega_{E/N} = 7.292\,115\,0 \times 10^{-5}\,\frac{\mathrm{rad}}{\mathrm{s}}$. Define the state vector as $\boldsymbol{x}(t) = \begin{bmatrix} \boldsymbol{r}(t) & \boldsymbol{v}(t) \end{bmatrix}^\top$.

(a) Linearize the system around a nominal trajectory $X_{\text{nom}}(t)$, deriving the system and output matricies $A = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ and $C = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}$.

(b) Write a function that outputs A_k.

(c) Write a function that outputs C_k.

(d) Numerically propagate (i) the nominal trajectory and (ii) the nominal trajectory perturbed an initial state deviation of $\delta\boldsymbol{X} = [30, 0, 0, 0, 0, 0.1]^\top$ for 90 minutes. Plot the difference in the position vector $\delta r(t)$ as a function of time.

(e) Propagate the state deviation directly using the integrated state transition matrix from the augmented nominal state vector (i.e. and plot the position vector deviation as a function of time.

(f) Increase the initial state perturbation to $\delta\boldsymbol{X} = [1000, 0, 0, 0, 0, 0.1]^\top$ and integrate (i) numerically and (ii) using the STM. Plot $\delta r(t)$ for both propagatation strategies.

(g) Explain what you see.

(h) A set of range measurements are provided for a different trajectory in `HW3-kepler-data.npy` from the same observer. Use the mesaurements to estimate the initial state deviation $\delta\boldsymbol{x}(t_0)$ using the same nominal trajectory as before.

(i) Explain your approach.

Solution

Part A

State: $\boldsymbol{x} = [\boldsymbol{r}; \boldsymbol{v}] \in \mathbb{R}^6$, dynamics

$$\dot{x} = \begin{bmatrix} \dot{r} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \\ -\frac{\mu}{\|r\|^3} r \end{bmatrix} =: f(x)$$

Let $r = \|\boldsymbol{r}\|$. The Jacobian is

$$\mathbf{A} = \frac{\partial f}{\partial x} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\ \mu\left(\frac{3rr^{\top}}{r^5} - \frac{\mathbf{I}_{3\times3}}{r^3}\right) & \mathbf{0}_{3\times3} \end{bmatrix}$$

Measurement: range to a ground observer $\rho = \|\boldsymbol{r} - \boldsymbol{r}_{\text{obs}}\left(\boldsymbol{t}\right)\|$
Hence

$$\mathbf{C} = \frac{\partial \rho}{\partial x} = \begin{bmatrix} \frac{(r - r_{\text{obs}}(t))^{\top}}{\|r - r_{\text{obs}}(t)\|} & \mathbf{0}_{1\times3} \end{bmatrix}$$

The observer position in ECI (with latitude ϕ, longitude λ, Earth radius R_E, rotation rate ω_E) is

$$r_{\text{obs}}^{\text{ECI}}(t) = \mathbf{R}_3\left(\omega_E t\right) \begin{bmatrix} R_E \cos\left(\phi\right)\cos\left(\lambda\right) \\ R_E \cos\left(\phi\right)\sin\left(\lambda\right) \\ R_E \sin\left(\phi\right) \end{bmatrix}, \quad \mathbf{R}_3(\theta) = \begin{bmatrix} \cos\left(\theta\right) & -\sin\left(\theta\right) & 0 \\ \sin\left(\theta\right) & \cos\left(\theta\right) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Part B

```
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   1.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  1.00000000e+00]
 [ 2.32419825e-06  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00 -1.16209913e-06  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00 -1.16209913e-06  0.00000000e+00
   0.00000000e+00  0.00000000e+00]]
```

Part C

```
[[ 0.59338974 -0.66973    -0.44648667  0.          0.          0.        ]]
```

Part D

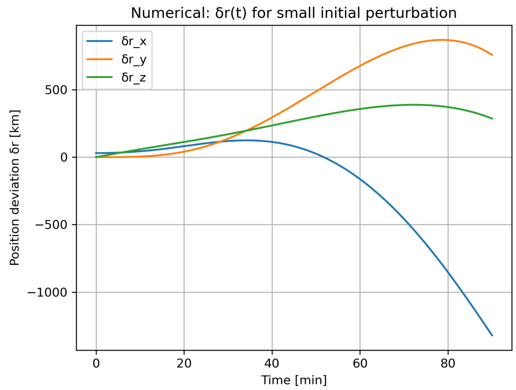

Figure 2: Numerical Integration d$\boldsymbol{X}$

Part E

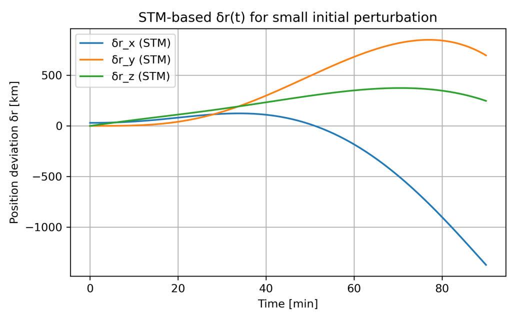

Figure 3: Analytic Integration d$\boldsymbol{X}$

Part F

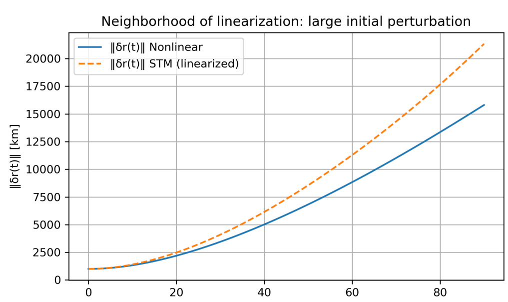

Figure 4: Critical d$\boldsymbol{X}_{\text{neighborhood}}$

Problem 3 continued on next page...

Part G

```
For small perturbations, the STM linearization matches the nonlinear δr(t) closely: the
direction and amplitude agree because the first-order model captures the local flow and gravity-gradient
coupling along the nominal path. When the initial deviation is increased, higher-order terms (curvature of
the vector field and coupling through 1/‖r‖^3) become significant over an orbital timescale. The STM keeps
projecting the initial deviation through Φ(t,t0) built on the nominal, so its prediction grows biased: phase
drifts and amplitude errors accumulate, especially near perigee where the field is most nonlinear.
The plot shows divergence in ‖δr(t)‖: nonlinear truth departs from the linear prediction, evidencing the
finite neighborhood of validity for the first-order model.
```

Part H

```
[ 7.38541948e+04  4.50585150e+06 -6.94356932e+06 -1.32578732e+03
 -1.42889210e+03  3.01612462e+03]
```

Part I

```
We linearize the range model about the nominal: y_k = ρ(r_k) with r_k ≈ r_nom,k + δr_k,
δX_k = Φ_k δX0. The measurement Jacobian is C_k = [ (r_nom,k - r_obs,k)^T / ρ_nom,k , 0_{1×3} ].
Thus y_k - ρ_nom,k ≈ (C_k Φ_k) δX0. Stacking all samples yields a linear least-squares problem:
min_δX0 || y_res - H δX0 ||_2, where rows of H are C_k Φ_k. We integrate once along the nominal
and its STM to get X_nom(t_k), Φ_k, build H and the residual vector y_res, then solve with
np.linalg.lstsq. This provides the best linearized estimate of the initial deviation δX0 given
range-only data and the chosen nominal trajectory.
```

Code

See the Python code for this assignment.